LIBRARY OF THE
UNIVERSITY OF ILLINOIS
AT URBANA-CHAMPAIGN
510.84
Il6r
no.100-110
cop.3

DIGITAL COMPUTER LABORATORY
UNIVERSITY OF ILLINOIS
URBANA, ILLINOIS

REPORT NO. 104

A METHOD FOR FACTORING THE ACTION OF ASYNCHRONOUS CIRCUITS

by W. D. Frazer and D. E. Muller

Presented at the AIEE Fall General Meeting
Chicago, Illinois, October 12, 1960. This
work was supported in part by the Office of
Naval Research under Contract Nonr-1934(27).

November 2, 1960

A METHOD FOR FACTORING THE ACTION OF ASYNCHRONOUS CIRCUITS

W. D. Frazer D. E. Muller

In this paper we shall discuss a procedure which enables one to analyze and trace the behavior of asynchronous circuits while testing certain properties of this behavior. Although methods for performing this type of analysis have been devised and used in the past, they have differed from the method described here in an important respect (1). In the present procedure information concerning the "topology" or interconnection structure of the circuit is used to simplify the analysis, while in earlier procedures this information was not used.

Let a circuit C be defined by a set of Boolean equations

$$z_1' = f_1(z_1, z_2, \ldots, z_n)$$
$$z_2' = f_2(z_1, z_2, \ldots, z_n)$$
$$\vdots$$
$$z_n' = f_n(z_1, z_2, \ldots, z_n)$$

Each variable z_i represents a signal or measurable binary quantity in the circuit, and each function f_j describes a logical element or gate in the circuit. Typical elements represented by such functions are the familiar AND, OR, and NOT elements of computer logic. The dependent variables z_i' are called the implied signals and represent the signal values toward which the given signals z_i are made to tend by the circuit logic.

While each function f_j is written as though it is a function of all n of the variables z_i, it is usually an explicit function of just a few of these n variables. The remaining variables which do not influence the functional value f_j may be optionally included or not among the arguments in any formal description of the circuit. It is precisely the information concerning explicit dependence which is used in the present analysis procedure.

To specify this explicit dependence, an n x n connection matrix, M, is constructed. Any given matrix element m_{ij} is made equal to 1 if the function f_j depends explicitly on z_i and is made equal to 0 otherwise. Thus, if f_j represents a two input AND element, m_{ij} will be made equal to 1 if and only if node i is one of the two inputs to this element.

An n-tuple $a = (a_1, a_2, \ldots, a_n)$ of specific values of the variables z_1, z_2, ..., z_n is called a state of the circuit. Therefore specifying the state of the circuit is equivalent to specifying the n signals in the circuit. With the passage of time the circuit will pass through a sequence of states

$$a(1), a(2), \ldots, a(k), \ldots \ .$$

In an asynchronous or unclocked circuit, the state sequence, in general, does not depend only on the initial state $a(1)$ and the logical properties of the circuit as described by the Boolean functions f_j. The state sequence also depends upon the relative speeds of the logical elements making up the circuit.

Nevertheless, a knowledge of the circuit logic permits one to determine a class of state sequences, called allowed sequences, such that the actual state sequence must be a member of this class. Specifically, one considers sequences in which each consecutive pair of states $a(k)$, $a(k + 1)$ bear either the relationship $a_i(k + 1) = a_i(k)$ or $a_i(k + 1) = a_i'(k)$ for each $i = 1, 2, \ldots, n$. Here $a_i'(k)$ is taken as the functional value $f_i(a_1(k), a_2(k), \ldots, a_n(k))$. Whenever $a_i'(k) = a_i(k)$, the i'th node is said to be in equilibrium and the value of $a_i(k + 1)$ is unambiguous. However, when $a_i'(k) \neq a_i(k)$ the i'th node is said to be excited and $a_i(k + 1)$ may assume either value; which value it actually assumes depends upon the speed of the i'th element relative to the other elements of the circuit.

One object of the analysis procedure is to determine whether or not the the circuit is speed independent in the sense that all allowed sequences starting with the given initial state $a(1)$ either terminate in the same state or same cycle of states. While a complete determination is not made, the stronger property of semi-modularity is checked. This property, also defined with respect to the initial state $a(1)$, is satisfied if and only if for each consecutive pair of states $a(k)$, $a(k + 1)$ in each allowed sequence starting with $a(1)$ and for each node i we have $a_i'(k) = a_i'(k + 1)$ whenever $a_i'(k) \neq a_i(k) = a_i(k + 1)$. Thus, a circuit is semi-modular with respect to $a(1)$ if no element in the circuit may pass from an excited to an equilibrium condition unless its signal changes.

To check semi-modularity each of the Boolean functions $a_j'(k) = f_j(a_1(k), a_2(k), \ldots, a_n(k))$ is computed for each given state $a(k)$. This functional value $a_j'(k)$ is then compared with $a_j(k)$, the corresponding signal value to determine whether they differ indicating that the element is excited. The state $a(k + 1)$ is now formed from $a(k)$ by changing the signal upon one of the excited nodes, say node i. Semi-modularity may be checked with respect to this transition by recomputing the functional values so as to form $a'(k + 1)$, and making certain that no nodes which were excited in state $a(k)$ have passed to equilibrium in state $a(k + 1)$ save node i, the node which underwent the change.

It is important to notice, at this point that only functions f_j which are known to depend explicitly upon the variable z_i need be checked during the transition described above. The other functions will not be affected and hence need not be checked, or even recomputed. Similarly, many state transitions a(k) to a(k + 1) need not be formed at all, for if all arguments of functions f_j depending explicitly on z_i had assumed the same values during some previously checked transition, the calculation need not be repeated. Thus, by using the information concerning explicit dependence as provided in the matrix M it is possible to avoid construction of many superfluous states and transitions.

Semi-modularity is checked whenever signal changes in two change paths occur on nodes i and j such that some function f_m exists which explicitly depends upon both z_i and z_j or such that either f_i depends explicitly upon z_j or f_j depends explicitly upon z_i. This testing is done whenever the two change paths in question represent node changes which may be taking place concurrently.

Determination of the explicit dependence described above is accomplished by use of the connection matrix M. For each change path form a vector v of 0's and 1's by the rule: $v_i = 1$ if node i undergoes a change in some transition a(k) to a(k + 1) in the change path in question, and $v_i = 0$ otherwise. Let vM represent the vector whose j'th component is $\sum_{i=1}^{n} v_i m_{ij}$, where Boolean union is used in the summation. It may therefore be seen that the j'th component of vM has the value 1 if and only if there is some node i, undergoing a change in the change path from which v was formed and such that f_j depends explicitly upon the corresponding variable z_i. Similarly, let $v M M^T$ represent the vector whose j'th component is $\sum_{i=1}^{n} \sum_{r=1}^{n} v_i m_{ir} m_{jr}$. Then the j'th component of $v M M^T$ is 1 if and only if there is function f_r which depends explicitly upon both z_i, z_j for some node i undergoing a change in the change path.

Thus, a systematic scheme for determining whether or not it is necessary to carry out a semi-modularity check may be formed by constructing the vectors v, vM, and $v M M^T$ for each change path.

In the proposed analysis procedure, change sequences are constructed according to the following rules:

a) In each state transition, a(k) to a(k + 1), just one node i undergoes a change.

UNIVERSITY OF
ILLINOIS LIBRARY

b) This node is chosen as one not excited in the previous state a(k -1), (all nodes are considered to be relaxed in the state which precedes the initial state) and one which has not changed during an earlier part of the sequence.

c) Beginning with the initial state, change sequences are formed according to rules (a) and (b) beginning with each, in turn, of the initially excited nodes. These sequences are terminated as outlined in rules (d) - (k). For each change sequence, an n-component vector V is recorded; V has the property that $v_i = 1$ if node i has changed in the sequence and $v_i = 0$ otherwise.

d) A test for semi-modularity is made for each transition.

e) After a change is made on a particular node k, in a change sequence C, a vector P is formed such that $p_k = 1$ and $p_i = 0$ for all $i \neq k$. The scalar $V M M^T P$ (where V is formed as above) is formed for each, in turn, of the existing previous change sequences, and for the preceding part of C.

f) If $V M M^T P = 0$ for all of the existing V, P is added to the V for C and another node to relax is chosen according to (b).

g) If $V M M^T P = 1$ for V corresponding to some sequence D, the terminal state of D is retained, but D is retraced up to its interaction with C giving D_1, an abbreviated version of D. A terminal state is formed by combining the terminal states of D_1 and C.

h) If at any point in the construction of a change sequence more than one new node becomes excited as a direct result of a given change, terminate the sequence.

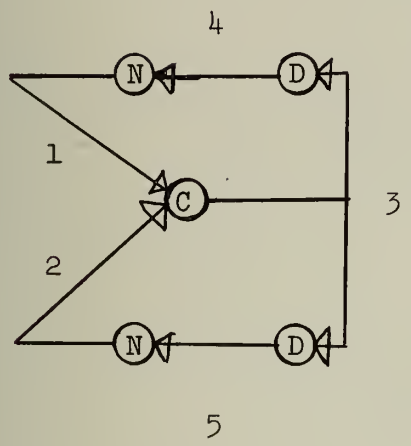
i) When any change sequence is terminated, the terminal state of that sequence and the V for that sequence are kept.

j) When all sequences originating from the given initial state have been pursued and all interaction tests completed, the initial state is discarded and the change sequence terminal states are treated as initial states.

EXAMPLE

Illustration of the Topological Technique

Consider the following circuit



Equations

$$z_1' = \bar{z}_4$$

$$z_2' = \bar{z}_5$$

$$z_3' = z_1 z_2 \vee z_1 z_3 \vee z_2 z_3$$

$$z_4' = z_3$$

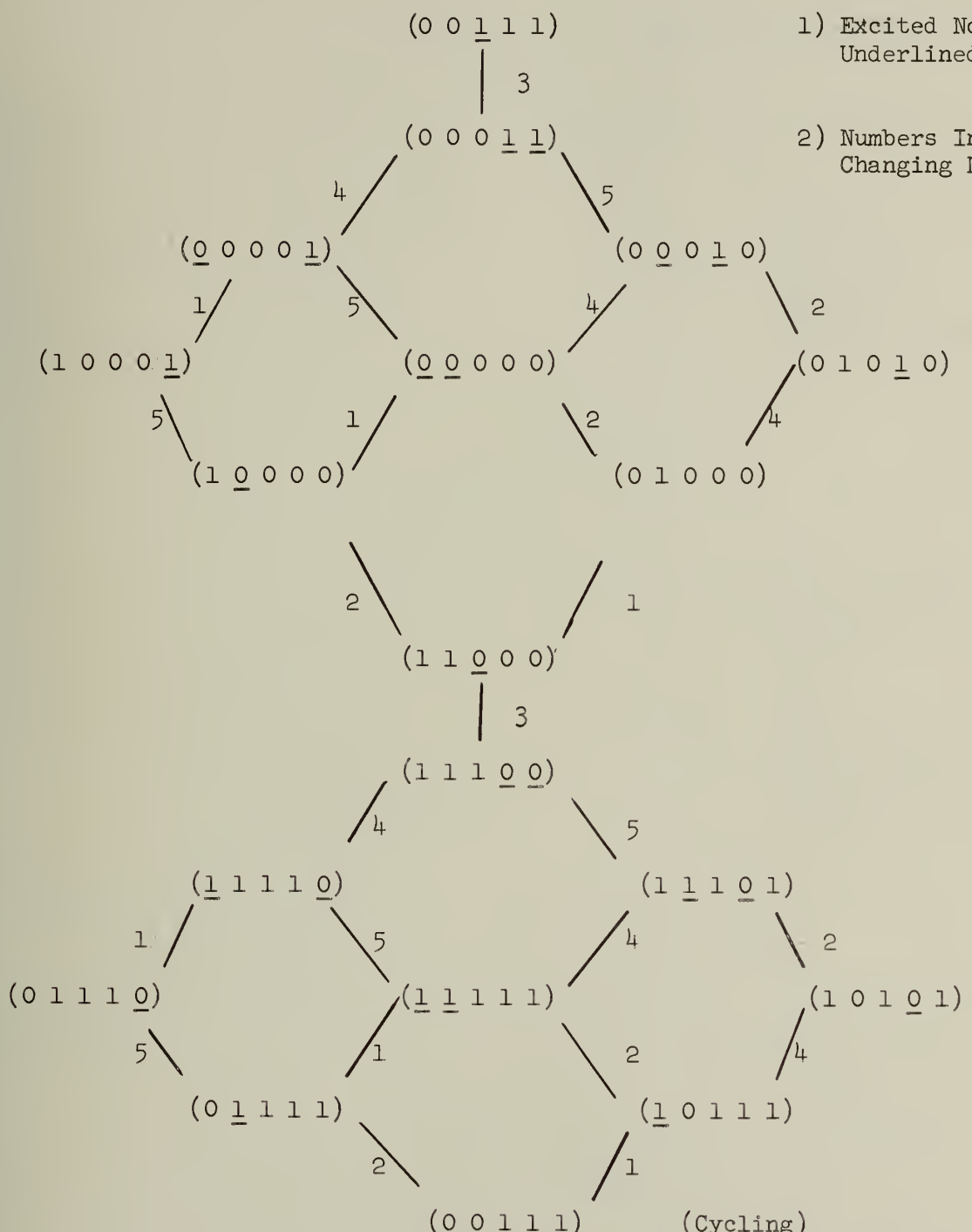
$$z_5' = z_3$$

$$M = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Initial State:

0 0 1 1 1

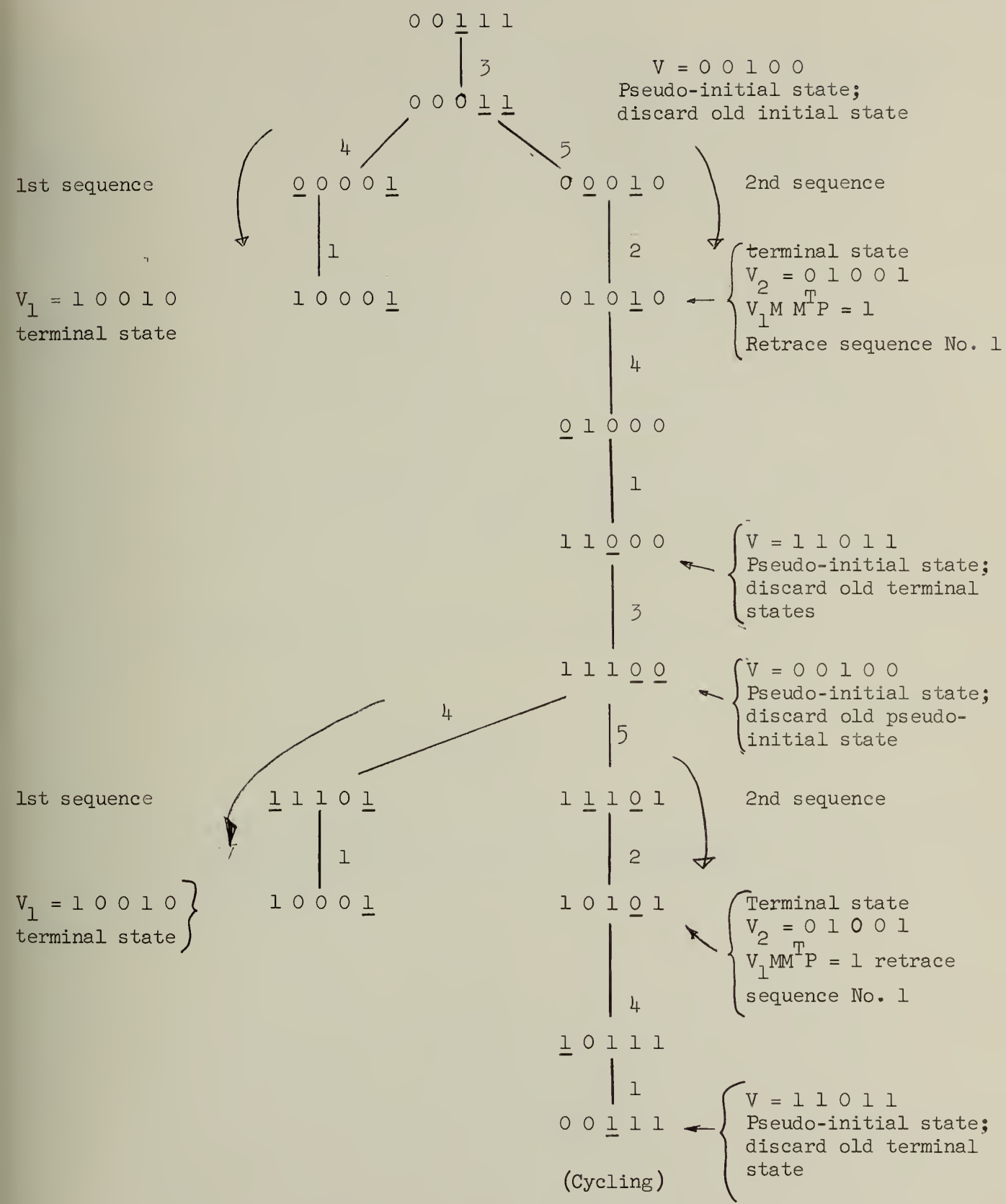
Analysis in which All States are Generated



1) Excited Nodes Underlined

2) Numbers Indicate Changing Nodes

EXAMPLE


0 0 1 1 1
3
0 0 0 1 1
V = 0 0 1 0 0
Pseudo-initial state;
discard old initial state
4
5
1st sequence
0 0 0 0 1
0 0 0 1 0
2nd sequence
1
2
terminal state
V2 = 0 1 0 0 1
V1 M MTP = 1
Retrace sequence No. 1
V1 = 1 0 0 1 0
terminal state
1 0 0 0 1
0 1 0 1 0
4
0 1 0 0 0
1
1 1 0 0 0
V = 1 1 0 1 1
Pseudo-initial state;
discard old terminal
states
3
1 1 1 0 0
V = 0 0 1 0 0
Pseudo-initial state;
discard old pseudo-
initial state
4
5
1st sequence
1 1 1 0 1
1 1 1 0 1
2nd sequence
1
2
V1 = 1 0 0 1 0
terminal state
1 0 0 0 1
1 0 1 0 1
Terminal state
V2 = 0 1 0 0 1
V1 MMTP = 1 retrace
sequence No. 1
4
1 0 1 1 1
1
0 0 1 1 1
V = 1 1 0 1 1
Pseudo-initial state;
discard old terminal
state
(Cycling)

CONCLUSION

The procedures outlined above and illustrated in the example lend themselves to use with high speed digital computers. A program for ILLIAC, the University of Illinois digital computer which performs analysis by generating all allowed sequences has been in use for some three years. It has been found to be too inefficient for use in large circuits. The procedure described in this paper should be most advantageous in just such cases.

One may gain some insight into the saving by considering the following example: Given two entirely independent circuits of m and n nodes, respectively, previous analysis requires the generation of, roughly $2^{(m+n)}$ states, while the topological method would require the generation of some number of states greater than m + n and less than $2^m + 2^n$ depending upon how complex the two circuits were.

The case worked out in the example illustrates in a crude fashion the way in which such savings come about. Even in a circuit as simple as this one, it is necessary to generate only about 3/4 as many states as would be generated by the old method.

The method proposed here is currently being implemented in the form of a program for ILLIAC.

REFERENCES

1. "An Illiac Program for Simulating the Behavior of Asynchronous Logical Circuits and Detecting Undesirable Race Conditions," W. S. Bartky and D. E. Muller (presented at the National Meeting of A.C. M., June, 1957) Digital Computer Laboratory File No. 221.

BOUND BY
HNM

UNIVERSITY OF ILLINOIS-URBANA
510.84 IL6R v.1 C002 v.100-110(1960-
Variations with size of characteristics
3 0112 088404055